Low Delay Audio Streaming for a 3D Audio Recording System

Marzena Malczewska, Tomasz Żernicki and **Piotr Szczechowiak**
Zylia Sp. z o. o.
Umultowska 85
Poznań, Poland,
{marzena.malczewska, tomasz.zernicki, piotr.szczechowiak}@zylia.pl

Abstract

This paper presents a prototype of a 3D audio recording system named AudioSense which uses Wireless Acoustic Sensors to capture spatial audio. The sound is recorded in real-time by microphones embedded in each sensor device and streamed to a Processing Unit for 3D audio compression. One of the key problems in systems which stream audio data is end-to-end latency. This paper is focused on analyzing a set of chosen parameters of the Opus codec in order to obtain the minimal delay. Experimental results on the prototype system have shown that it is possible to achieve below 10ms of end-to-end audio delay with the use of the Opus codec.

Keywords

audio streaming, opus codec, low latency streaming, wireless sensor network, spatial audio

1 Introduction

The area of 3D audio and object based audio is currently a hot research topic as evidenced by a large number of research papers and new emerging standards such as MPEG-H 3D Audio. The majority of the research efforts in this area are concentrated on the audio processing and rendering side. The problem of 3D audio recording is getting less attention in the literature.

Channel-based method of spatial sound production assume that the number of microphones used during the recording is directly proportional to the number of loud speakers used during sound rendering. This can lead to large numbers of microphones in case of multiple audio channels recordings. In addition, proper setting and tuning of microphones in the field can be a tedious task which requires many resources in terms of time and manpower. Current distributed recording systems use wired microphones, which makes it difficult to deploy and use the system in certain environments.

To solve the limitations of current spatial audio recording systems, the AudioSense system is being developed. It introduces object-based sound representation and wireless audio streaming. The system can be described as a Wireless Acoustic Sensor Network (WASN) [Bertrand, 2011; Akyildiz et al., 2007]. In this system individual sound sources (objects) are extracted from a sound mixture using sound source separation techniques (e.g Independent Component Analysis [Comon, 1994]). Object based audio representation gives high flexibility in terms of sound rendering (easy rendering for headphones, stereo, 5.1, 7.1 systems) and enables interactive manipulation of individual sound objects during playback.

The proposed AudioSense system has many possible applications and can be used for both indoor and outdoor audio recordings. The system can be used in teleconference applications to add the possibility to identify speakers by speech direction. It can be also used for wildlife monitoring and live TV broadcasts from the field. The AudioSense technology has applications in surveillance systems where it can identify and track objects based on sound processing. The system can be also applied in the entertainment industry in case of movies, games and virtual reality applications that require immersive 3D sound.

Realisation of the AudioSense system is a challenging task that requires overcoming major challenges in such areas as audio streaming, audio coding, sound sources separation and audio synchronisation. This paper is focusing on designing a low delay audio streaming mechanism that meets the strict requirements of the AudioSense system.

The AudioSense system consists of battery operated devices with low processing capabilities. Therefore audio recording, coding and streaming has to be performed with energy efficiency in mind. Live applications of the system require also low latency audio streaming that is reliable and allows simultaneous streaming of data from multiple devices over the wireless

medium.

In order to minimise the end-to-end delay it is necessary to optimise the audio recording process, use a low latency audio codec and streaming method. This paper shows the design of the system that tries to achieve this goal. It presents the technologies and design choices made to implement a low delay audio streaming system on off-the shelf embedded devices. The results achieved during the performance evaluation of the system show that it is possible to achieve a low end-to-end delay of 10ms for wireless audio streaming within the AudioSense system.

The rest of the paper is organised as follows. Section 2 presents the related work in the area of spatial audio recording systems. Section 3 describes the architecture of the AudioSense system together with hardware and software components implemented to build the first prototype of the system. The results achieved during the performance evaluation phase are presented in Section 4. Finally Section 5 concludes the paper and describes the lessons learnt from implementing a low delay audio streaming system.

2 Related work

Spatial audio recording systems are gaining on popularity with the introduction of 3D audio systems and technologies that can reproduce truly immersive sound. Majority of existing systems for spatial audio recording use wired microphones [Gallo et al., 2007] to capture the virtual sound stage. This fact limits drastically the mobility of such systems and increases significantly their deployment time.

In the literature one can find also wireless systems for distributed audio recording like [Taysi et al., 2010] and [Pham et al., 2014]. The main problem with such systems is that the wireless sensor network devices are equipped with low quality microphones, amplifiers and A/D converters due to the low cost and high energy efficiency of the system. Sounds recorded with such systems have insufficient quality for many audio applications.

One of the systems that tries to overcome the problems of low cost WASNs is WiLMA [Schörkhuber et al., 2014]. The system introduces a wireless microphone array that offers high quality audio recording and processing. WiLMA enables connection of up to 4 professional microphones to each sensor module and provides wireless synchronisation for audio recordings. Similar approach to system design is presented also in [Mennill et al., 2012] where a distributed microphone array system is used for environmental monitoring and animals recording. This system is also based on battery operated sensors and uses GPS for accurate synchronisation of the recordings.

One of the limitations of spatial audio recording systems presented in [Mennill et al., 2012] is that the system does not offer continuous real-time wireless audio streaming. All the recordings are stored on local flash memory of the sensor devices. The AudioSense system takes the next step in spatial audio recording systems by providing low delay wireless streaming capabilities and audio representation in the object-based format. These features open up the door for a whole new range of audio applications that can be realised with the use of the AudioSense system.

3 System overview

The proposed architecture of the AudioSense system is presented in Figure 1. From the functional side the system can be divided into two parts. The first part consists of Acoustic Sensors that form a wireless network responsible for audio recording. The second part includes an embedded device which performs 3D audio processing. Each device in the wireless sensor network has one or several microphones, A/D converter and performs initial audio compression. Compressed audio signals are transmitted through the Gateway to the Processing Unit. The Gateway serves as an interface between the wireless and the wired part of the system. After reception of the audio signals the Processing Unit performs aggregation of the individual streams.

Each of the streams is decoded and synchronised with each other. In the next step the process of sound sources separation is performed to generate individual audio objects [Salaün et al., 2014], [Ozerov et al., 2012]. These objects are then used in the process of 3D audio coding (e.g. MPEG-H 3D Audio [ISO/IEC WD 23008-3, 2014]. Finally the encoded audio is transmitted over the Internet to the client side where the sound rendering is performed.

3.1 Hardware components

From the hardware perspective the Acoustic Sensor prototype consists of a Beaglebone Black [Coley, 2014] with an Audio Cape board [BeagleBoard, 2012] and a dedicated microphone

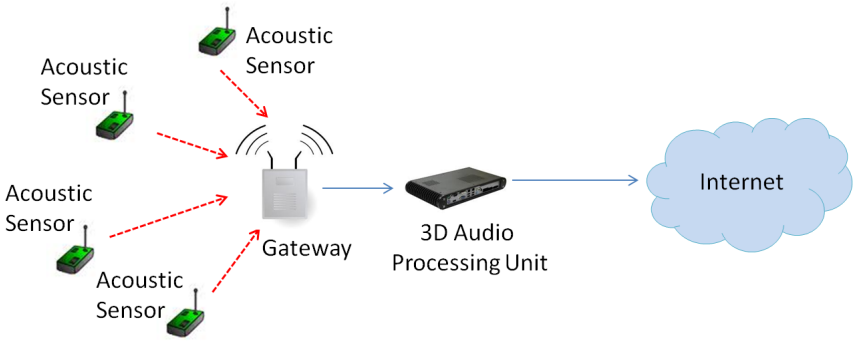

Figure 1: Architecture of the AudioSense system.

board designed in-house. Beaglebone Black is a low-power single board computer based on 1GHz ARM Cortex A8 CPU. The board provides only one 12-bit analog-to-digital converter which is not sufficient for any professional audio applications. Hence the usage of an Audio Cape (6 channels of up to 96 kHz sampling at 24 bit) is required to improve the quality of the recorded sound. For the microphone board a pair of Monacor MCE-4000 electret omnidirectional microphones is selected due to high signal to noise ratio and very good sensitivity. Each microphone is connected to a low noise operational amplifier - MCP6021. The amplified acoustic signal is passed on to the Audio Cape where analog to digital conversion is executed. Next, the digital data in one of available formats (e. g. S16LE) is sent to the Beaglebone Black. Each acoustic sensor is equipped also with a wireless interface compatible with the IEEE 802.11 a,b,g,n standards. The first version of the prototypical Acoustic Sensor is presented in Fig. 2.

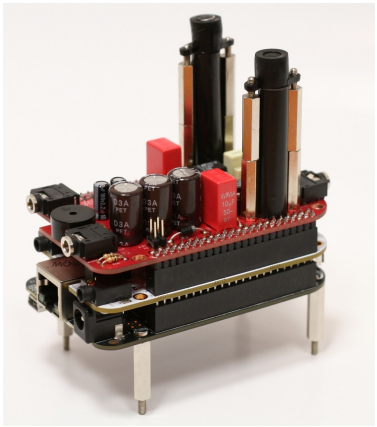

Figure 2: First version of the Acoustic Sensor prototype.

3.2 Software components

The prototype of Audio Sensor is running Debian Jessie Linux with kernel version 3.8.13. In order to implement audio processing on the device, the Gstreamer framework is utilised. Sound capturing, coding and streaming are all implemented within a single Gstreamer v1.4 pipeline. Figure 3 illustrates Gstreamer pipelines implemented on both the Acoustic Sensor and the 3D Audio Processing Unit.

The pipeline on the Acoustic Sensor side is responsible for capturing audio samples using the ALSA plugin and encoding them with the Opus [Valin et al., 2013] encoder. Next, every packet is encapsulated in the RTP packet and sent via UDP to the Processing Unit.

On the Processing Unit side each received packet is processed by the depayloader and Opus decoder. This processing is performed for each stream independently. Next step is the separation plugin, which gets n streams and, after performing the process of sound sources separation, generates m audio objects. The processed data is passed on to the multiqueue and then interleaved to form a multichannel wave file. For this purpose a new Gstreamer module is implemented called *WavNChEnc*. The stream generated by the module is then passed to the MPEG-H 3D Audio codec which generates a single .mp4 file.

To measure time of encoding, the measurements points were set right before and after Opus encoder. Respectively, on the Processing Unit the points were set before and after the Opus decoder. Streaming time was measured with the measurement points set just before UDP transceiver in Acoustic Sensor and right after RTP depayloader in the Processing Unit.

One of the key aspects in networked audio

systems is audio synchronisation. In order to provide synchronisation of the recorded audio streams, a separate synchronisation module is implemented. The synchronisation method applied is a hybrid approach based on reference broadcast that uses the ideas presented in [Elson et al., 2002] and [Budnikov et al., 2004]. Using this hybrid synchronisation method it is possible to achieve a synchronisation error of around 200μs.

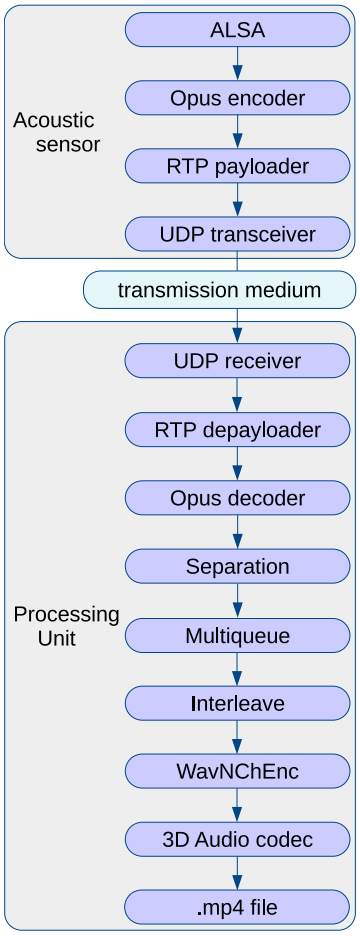

Figure 3: Gstreamer pipelines implemented on the Acoustic Sensor and the 3D Audio Processing Unit side.

4 Performance evaluation

4.1 Test scenarios

The performance of the designed 3D Audio recording system was evaluated using several test scenarios. The main goal of the experiments was to adjust and optimise hardware and software components of the system to achieve minimal streaming delay for different audio bitrates and network setups. Audio streaming latency was measured in an end-to-end manner which includes:

- Audio encoding time - time needed to encode one whole buffer of data by the Opus encoder. Such measurements were executed for different codec parameters which have the highest impact on the encoding time (e.g. bitrate, complexity, frame-size).
- Audio decoding time - measurement of decoding time for the same set of parameters as in the case of audio encoding.
- Audio transmission time - packets latency measurement when streaming wirelessly over Wi-Fi (IEEE 802.11n).

For the Audio streaming tests the Opus parameters were constant while the network setup was different in each experiment. The system was tested with several Acoustic Sensors in the network. In each of the cases the distance between the Acoustic Sensors and the 3D Audio Processing Unit was different to test the system in different working conditions. In addition to latency tests the experiments included also CPU usage measurements for Opus encoding and decoding.

Impact of selected parameters of the Opus codec on the quality of sound was not the subject of our tests. Several tests were performed in the past and are well described in [Hoene et al., 2011].

4.2 Results

This subsection presents experimental results achieved by measuring the end-to-end audio streaming delay in the AudioSense system. The first set of tests was performed to measure the encoding delay of the Opus codec in order to find the optimal codec parameters that enable minimal processing latency. Three parameters of the codec were identified as possible candidates for processing delay optimisation:

- *Complexity* - is defined as a trade-off between processing complexity and quality/bitrate. This parameter is selected using an integer from 0 to 10, where 0 is the lowest complexity and 10 is the highest. In the experiments fixed values of 0, 3, 6 and 10 were used to check what is the influence of complexity on the processing delay.
- *Frame size* - Opus has fixed frame durations of 2.5, 5, 10, 20, 40, and 60 ms. Increase in the frame duration has influence

on coding efficiency improvement but the gain becomes small for frame sizes above 20 ms.

- *Bitrate* - Opus supports different bitrates in the range between 6 kbit/s and 510 kbit/s. Higher bitrate results in higher quality audio and lower latency in packets delivery at the cost of increased bandwidth.

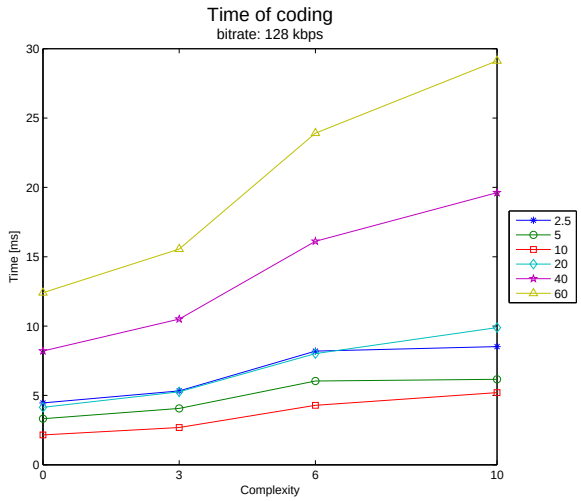

Figure 4: Opus encoding time for different values of *complexity* and *frame-size*.

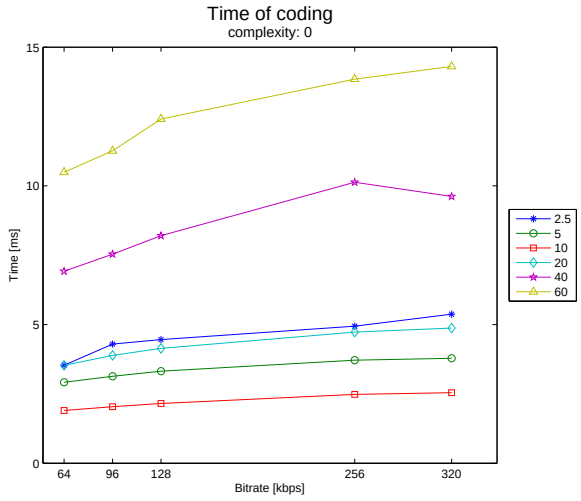

Figure 5: Opus encoding time for different values of *bitrate* and *frame-size*.

Figure 4 shows Opus encoding delay for different complexity and frame size settings (different colors on the figure correspond to different frame sizes). For all measurements the bitrate remained constant at the level of 128 kbit/s. It is clearly visible that the average encoding latency increases with higher complexity values. The difference is especially visible for higher frame durations of 40 and 60 ms where the latency is 3 to 6 times higher than in the case of smaller frame sizes. Therefore the best values of frame size in case of the AudioSense system are below 20ms where the encoding delay is smaller than 10ms. Surprisingly the frame size of 2.5ms is providing a similar encoding delay as in the case when the frame duration is set to 20ms. In terms of complexity the optimal value is 3 with frame size of 10 ms.

The influence of different audio bitrates on the Opus encoding delay is illustrated in Figure 5. The complexity parameter in all cases is fixed at 0. The experiments are performed for five audio bitrates (64, 96, 128, 256 and 320 kbit/s) and the same frame size values as in the previous test. The graph shows that significant increase in processing time is visible for larger values of frame size (40 and 60 ms). It is evident that the bitrate change has much smaller effect on encoding time than the change of the complexity parameter. For frame size values below 20ms the change of bitrate has very small effect on the encoding delay - only 1ms increase when changing the bitrate from 64kbit/s to 320 kbit/s. This experiment shows once again that the frame size of 10ms provides the optimal setting in terms of Opus encoding latency.

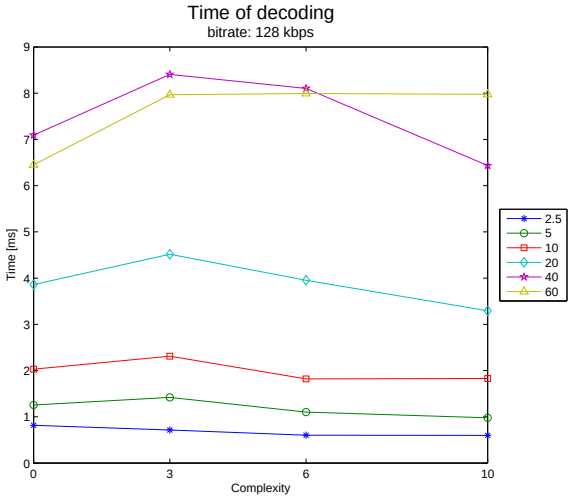

Figure 6: Opus decoding time for different values of *complexity* and *frame-size*.

Opus decoding latency is tested in a similar manner as in the case of encoding. Figure 6 shows the impact of the complexity parameter on the decoding time for different frame duration values. The audio bitrate is fixed at 128 kbit/s. As can be seen in Figure 6 the com-

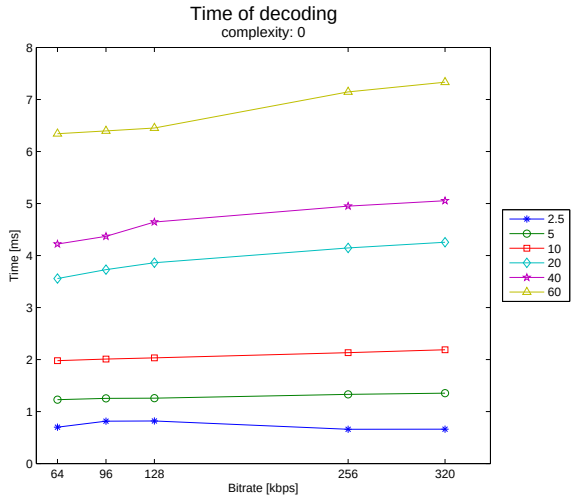

Figure 7: Opus decoding time for different values of *bitrate* and *frame-size*.

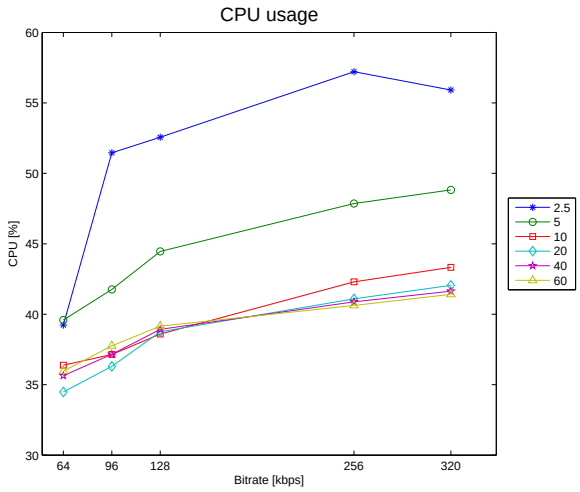

Figure 8: CPU usage for different values of *bitrate* and *frame-size*.

plexity parameter has a small impact on the overall audio decoding time. For smaller values of frame size (10ms and below) the decoding time remains the same for different complexity values. The difference is visible only in case of larger frame size values (20, 40 and 60ms) where the decoding delay can increase or decrease by around 1ms with the change of codec complexity.

Figure 7 presents Opus decoding times with respect to different audio bitrates. In all cases the complexity parameter is set to 0. It is clearly visible that audio bitrate has very small impact on the decoding time. The main parameter that has the biggest influence on decoding time is frame duration. From the point of view of Opus decoding, the best performance in terms of execution time can be achieved for the smallest possible values of frame size: 2.5 and 5ms.

The AudioSense system consists of battery operated sensor devices therefore the power consumption during codec operation is an important parameter that can limit the total operation time of the system. Figure 8 presents the CPU usage on the Acoustic Sensor while performing coding and decoding using the Opus codec. The measurements are performed for six different audio bitrates and six frame durations. In all cases the CPU operation remains between 35% and 57%. Highest CPU usage is recorded for the smallest frame size value (2.5ms). For all frame sizes between 10ms and 60ms the CPU usage stays on the same levels. The influence of audio bitrate on CPU processing is not significant as changing the bitrate from 64 kbit/s to 320 kbit/s increases the CPU usage by 7% on average. The complexity parameter of the Opus codec has a stronger influence on the CPU processing than audio bitrate change. Changing the complexity from 0 to 3 increases the CPU usage by 10% on average. Switching from 3 to 6 adds another 10% of CPU processing. It is recommended to set the complexity on 3 or lower in order to keep the CPU usage below the level of 50%. From the energy efficiency point of view the optimal frame size is equal to 10ms.

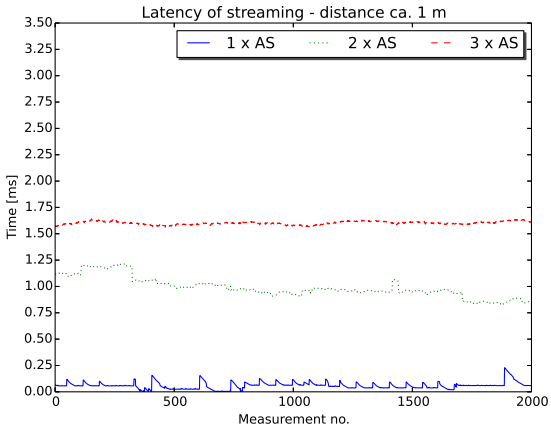

Figure 9: Streaming delay measurements with a distance of 1m between devices.

Audio encoding and decoding adds a significant delay in the audio processing pipeline of the AudioSense system. The third factor that adds an additional delay is audio streaming over the wireless channel. In order to mea-

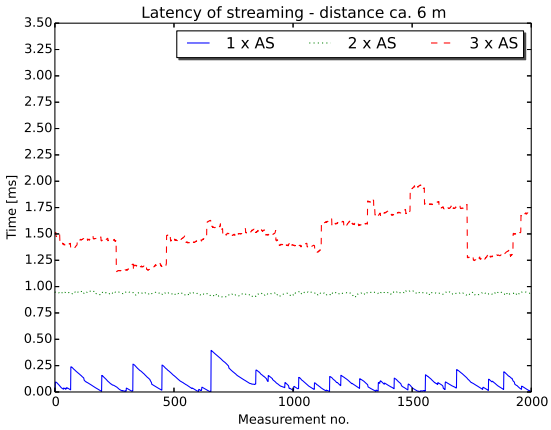

Figure 10: Streaming delay measurements with a distance of 6m between devices.

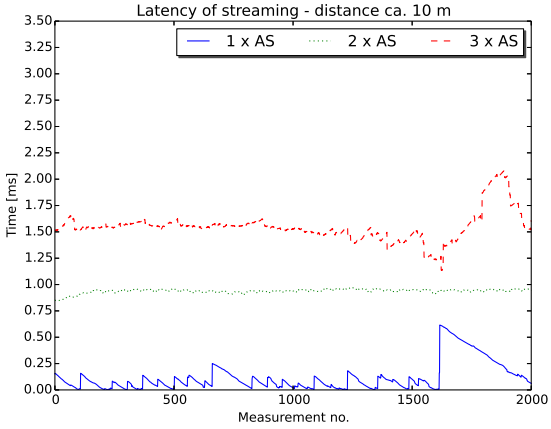

Figure 11: Streaming delay measurements with a distance of 10m between devices.

sure streaming latency over Wi-Fi several experiments are performed using the prototype AudioSense system.

All the tests are made with the same parameters of the Opus codec: sampling rate 48kHz, bitrate 128 kbit/s, complexity 0, frame size 10ms. Figure 9 presents the first set of experiments where the network consists of one, two or three Acoustic Sensors (AS). In all cases the distance between the 3D Audio Processing Unit and Acoustic Sensors is equal to 1m. The measurements are taken over 2000 audio samples. The streaming is performed under Line of Sight (LOS) conditions using the 802.11 n mode. It is clearly visible in Figure 9 that the streaming delay is the lowest (around 70μs) when there is only one Acoustic Sensor in the network. Addition of the second sensor that sends simultaneously audio data to the processing unit increases significantly the overall packets delivery time to around 1ms on average. The network with three Acoustic Sensors increases the delay even further to around 1.6ms.

Figures 10 and 11 show the results of the same experiment as above but under different network conditions. The distance between the devices is increased to 6m and 10m respectively. The streaming is performed under Non Line of Sight (NLOS) conditions. The results demonstrate that the increase in distance between devices has small influence on the average audio streaming delay. The average delay for packet reception remains at the same levels in all three sets of tests. The main difference can be noticed in the jitter levels which are much higher when using the system in NLOS conditions.

5 Conclusions

This paper presents the architecture of the AudioSense system which is designed to record and process spatial audio. All the hardware and software components of the prototype implementation of the system are described in detail. The result of the work is a wireless acoustic sensor network capable of distributed sound recording in an object-based audio format.

The paper focuses also on the development of a low delay audio streaming technique which meets the strict requirements of the AudioSense system. For this purpose the Gstreamer framework is utilised together with the Opus codec. The optimal working parameters for the codec are selected through experimental evaluation and the end-to-end delay is measured for different setups of the wireless network. The results demonstrate that it is possible to achieve an average delay below 10ms for coding, transmission and decoding of the audio signal in a wireless system of several Acoustic Sensors.

For the future work it would be interesting to test the system on a larger scale with parallel transmissions from many Acoustic Sensors. The capacity of the system and transmission delay can be further optimised by utilising wireless streaming in the 802.11 ac standard. For the needs of sound sources separation it will be beneficial to apply a hardware based synchronisation method which would limit the synchronisation error to several μs.

6 Acknowledgements

This work was supported by National Centre for Research and Development (NCBiR), Poland,

"Leader" programme.

References

Adapteva, 2014. *Parallella Reference Manual 13.11.25*.

Ian F Akyildiz, Tommaso Melodia, and Kaushik R Chowdury. 2007. Wireless multimedia sensor networks: A survey. *Wireless Communications, IEEE*, 14(6):32–39.

BeagleBoard, 2012. *BeagleBone Audio Cape Revision A1 System Reference Manual*, October.

Alexander Bertrand. 2011. Applications and trends in wireless acoustic sensor networks: a signal processing perspective. In *Communications and Vehicular Technology in the Benelux (SCVT), 2011 18th IEEE Symposium on*, pages 1–6. IEEE.

D. Budnikov, I. Chikalov, S. Egorychev, I. Kozintsev, and R. Lienhart. 2004. Providing common i/o clock for wireless distributed platforms. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*, volume 3, pages iii–909–12 vol.3, May.

Gerald Coley, 2014. *BeagleBone Black System Reference Manual - Revision B*, January.

Pierre Comon. 1994. Independent component analysis, a new concept? *Signal processing*, 36(3):287–314.

Jeremy Elson, Lewis Girod, and Deborah Estrin. 2002. Fine-grained network time synchronization using reference broadcasts. In *Proceedings of the 5th Symposium on Operating Systems Design and implementation*, OSDI '02, pages 147–163, New York, NY, USA. ACM.

Emmanuel Gallo, Nicolas Tsingos, and Guillaume Lemaitre. 2007. 3d-audio matting, post-editing and re-rendering from field recordings. *EURASIP: Journal on Advances in Signal Processing*. Special issue on Spatial Sound and Virtual Acoustics.

Christian Hoene, Jean-Marc Valin, Koen Vos, and Jan Skoglund. 2011. Summary of opus listening test results. Technical report, IETF.

ISO/IEC WD 23008-3. 2014. Information technology - High efficiency coding and media delivery in heterogeneous environments - Part 3: 3D audio. Technical report, International Organization for Standardization / International Electrotechnical Commission, International Telecommunications Union – Telecommunication, January.

Daniel J Mennill, Matthew Battiston, David R Wilson, Jennifer R Foote, and Stephanie M Doucet. 2012. Field test of an affordable, portable, wireless microphone array for spatial monitoring of animal ecology and behaviour. *Methods in Ecology and Evolution*, 3(4):704–712.

Alexey Ozerov, Emmanuel Vincent, and Frédéric Bimbot. 2012. A General Flexible Framework for the Handling of Prior Information in Audio Source Separation. *IEEE Transactions on Audio, Speech and Language Processing*, 20(4):1118 – 1133, May. 16.

Congduc Pham, Philippe Cousin, and Arnaud Carer. 2014. Real-time on-demand multi-hop audio streaming with low-resource sensor motes. In *Local Computer Networks Workshops (LCN Workshops), 2014 IEEE 39th Conference on*, pages 539–543. IEEE.

Yann Salaün, Emmanuel Vincent, Nancy Bertin, Nathan Souviraà-Labastie, Xabier Jaureguiberry, Dung T. Tran, and Frédéric Bimbot. 2014. The Flexible Audio Source Separation Toolbox Version 2.0. May.

Christian Schörkhuber, Markus Zaunschirm, and IO-hannes Zmölnig. 2014. Wilma-wireless largescale microphone array. In *Linux Audio Conference*, volume 2014.

Z. Cihan Taysi, M. Amac Guvensan, and Tommaso Melodia. 2010. Tinyears: Spying on house appliances with audio sensor nodes. In *Proceedings of the second ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, BuildSys '10, pages 31–36, New York, NY, USA. Association for Computing Machinery.

Jean-Marc Valin, Gregory Maxwell, Timothy B Terriberry, and Koen Vos. 2013. High-quality, low-delay music coding in the opus codec. In *Audio Engineering Society (AES) Convention no. 135*. Audio Engineering Society.